TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

Tutkintotyö

Johannes Pystynen

SYMBIAN C++, PYTHON JA MIDP JAVA S60-SOVELLUSALUSTALLA

Työn valvoja: Tony Torp
Työn teettäjä: Teleca Finland Oy
Tampere 2007

TAMPEREEN AMMATTIKORKEAKOULU

Työn tekijä: Johannes Pystynen
Työn nimi: Symbian C++, Python ja MIDP Java S60-sovellusalustalla

Päivämäärä: 2.5.2007
Sivumäärä: 38 sivua ja 2 liitesivua
Hakusanat: Symbian, C++, S60, Python, PyS60, MIDP Java
Koulutusohjelma: Tietotekniikka
Suuntautumisvaihtoehto: Ohjelmistotekniikka

Työn valvoja: Tony Torp
Työn teettäjä: Teleca Finland Oy

TIIVISTELMÄ

Nokian S60 on yksi yleisimmistä Symbian OS -käyttöjärjestelmän sovellusalustoista. S60-alustalle tarjolla olevien työkalujen ja ohjelmointikielten joukko on kasvanut sitä mukaa, kun alustaa on kehitetty. Lisäksi Symbian OS -käyttöjärjestelmälle tehtävää natiivikehitystä on pidetty yleisesti haastavana. Näiden seurauksena ohjelmistosuunnittelijat ovat pyrkineet löytämään yhä tehokkaampia keinoja sovelluskehitykseen.

Tässä työssä vertailtiin Symbian C++:n, Pythonin ja MIDP Javan käyttöä ohjelmistokehityksessä S60-sovellusalustalla. Työssä tutkittiin sovelluskehityksen näkökulmasta sitä, mihin tilanteisiin kukin näistä kehitysympäristöistä sopii, sekä pyrittiin kartoittamaan ympäristöjen mahdollisia etuja toisiinsa nähden eri osa-aluilla. Työ pyrki myös kartoittamaan sitä, kuinka paljon kullakin kehitysympäristöllä tehtävä sovelluskehitys vie aikaa.

Työssä lähestyttiin eri kehitysympäristöjä lähdemateriaalin ja käytännön kautta. Eri ohjelmointikielillä tehtävää kehitystä tarkasteltiin koodiesimerkein. Ohjelmointikieliä tutkittiin myös niiden virheiden ja poikkeusten käsittelyn avulla. Kehitysympäristöjen työkaluja pyrittiin vertailemaan käytännön testien avulla.

Tutkimuksen pohjalta nousi esiin eri kehitysympäristöjen vahvuuksia ja heikkouksia. Käyttöliittymäpohjaiseen sovelluskehitykseen Java-työkalut näyttivät soveltuvan näistä kolmesta parhaiten, kun taas Python näytti tarjoavan mahdollisuuden nopeaan ja tehokkaaseen pienten komponenttien sovelluskehitykseen. Symbian C++:lla tehtävään sovelluskehitykseen oli puolestaan tarjolla paljon hyviä esimerkkikoodeja ja laaja SDK:n mukana tuleva apu, jotka molemmat auttavat muuten vaativassa Symbian C++ -kehityksessä. Tutkimustulosten perusteella voidaan todeta, että varsinkin pienten sovellusten, kuten demojen, kehittämiseen Python ja Java tarjoavat erittäin hyvän vaihtoehdon perinteiselle Symbian C++ -kehitykselle.

TAMPERE POLYTECHIC

Author:	Johannes Pystynen
Name of the thesis:	Symbian C++, Python and MIDP Java on S60 Platform
Date:	02/05/2007
Number of pages:	38 pages and 2 appendices
Keywords:	Symbian, C++, S60, Python, PyS60, MIDP Java
Degree programme:	Computer systems engineering
Specialisation:	Software Engineering
Thesis supervisor:	Tony Torp
Comissioning company:	Teleca Finland Oy

ABSTRACT

The S60 software platform is currently amongst the leading smartphone platforms in the world. It is a platform for mobile phones that uses the Symbian operating system. S60 platform tools, and the number of supported programming languages, have increased during the platform's development.

This thesis evaluates the use of Symbian C++, Python and MIDP Java in software development on the S60 platform. Research has been done to find possibilities that these different environments and programming languages have to offer.

This thesis concentrates to these different development environments with different sources and with practice. Different programming languages were under examination with code examples, and they were also evaluated by their error and exception handling. The development tools were evaluated by their use and easiness.

UI based software development is at the moment easiest to do with Java tools. Python seems to offer the best way to do software components very quickly. Software development with Symbian C++ seems to benefit from good programming examples, and great SDK help. The results were that at least the smaller software components and demos could be done with Python or Java.

ALKUSANAT

Tämä insinöörityö on tehty talven 2006 ja kevään 2007 aikana. Työn ohjaajana on toiminut Ilkka Korhonen Teleca Finland Oy:stä ja valvojana Tony Torp Tampereen ammattikorkeakoulusta.

Kiitokset esimiehelleni työn tekemiseen annetusta mahdollisuudesta ja ajasta. Erityiskiitokset muutamille työkavereilleni, jotka ovat auttaneet aina tarvittaessa. Kiitokset myös työn ohjaajalle kiinnostavasta aiheesta.

Kiitokset Johannalle ja muulle perheelleni tärkeästä tuesta läpi opiskelun. Kiitokset myös kaikille kavereilleni, joiden avulla olen päätynyt tähän pisteeseen.

Tampereella 2.5.2007

_____

Johannes Pystynen

SISÄLLYSLUETTELO

LYHENTEET JA TERMIT

ANSI	American National Standards Institute.
API	Application Programming Interface. Ohjelmointirajapinta.
CLDC	Connected limited device configuration. Yhteydellisen vaatimattoman laitteen konfiguraatio.
DLL	Dynamic Link Library. Jaettu kirjasto.
EKA	Epoc Kernel Architecture.
GPL	General Public License. GNU, yleinen lisenssi.
GPS	Global Positioning System.
J2ME	Java 2 Platform, Micro Edition.
J2SE	Java 2 Standard Edition.
JLS	Java Language Standard.
JVMS	Java Virtual Machine Standard.
MIDP	Mobile Information Device Profile.
OBEX	OBject EXchange.
OS	Operating System. Käyttöjärjestelmä.
PDA	Personal Digital Assistant. PDA (kämmenmikro).
SDK	Software Development Kit.
S60	Nokian ohjelmistoalusta Symbian OS -käyttöjärjestelmän päällä.
UIQ	User Interface Quartz. Ericssonin ohjelmistoalusta Symbian OS -käyttöjärjestelmän päällä.

1 JOHDANTO

Tässä työssä on tutkittu Symbian C++-, Python- ja MIDP Java -kehitysympäristöillä toteutettavaa ohjelmistokehitystä Nokian S60-alustalla. Työ on toteutettu Teleca Finland Oy:lle talven 2006 ja kevään 2007 aikana. Työn rakenne seurailee Tampereen ammattikorkeakoulun tekniikan ja metsätalouden koulutusohjelmien tutkintotyöohjetta.

Johdannon jälkeisissä luvuissa esitellään ensiksi Symbian OS -käyttöjärjestelmää, S60-sovellusalustaa, MIDP Javaa ja Pythonia. Tämän jälkeen tutustutaan lähemmin C++-, Java- ja Python-kehitysympäristöjen tarjoamiin työkaluihin, joiden avulla S60-sovelluskehitys onnistuu. Työn loppupuolella paneudutaan vielä tarkemmin näihin eri ohjelmointikieliin koodiesimerkein. Lopuksi pohditaan eri kehitysympäristöjen käyttömahdollisuuksia sovelluskehityksessä ja pyritään löytämään jokaiselle tehokkaimmat käyttökohteet.

1.1 Toimeksianto

Teleca Finland Oy on keskittynyt mobiililaitteiden ohjelmistokehitykseen muun muassa S60-alustalla. Tarjolla olevien kehitysympäristöjen lisääntyessä yrityksen on tärkeää löytää tehokkaimmat ratkaisut eri sovellusten kehitykseen. Tästä johtuen työni aiheeksi muodostui eri kehitysympäristöillä tehtävän sovelluskehityksen vertailu S60-sovellusalustalla.

1.2 Tavoitteet

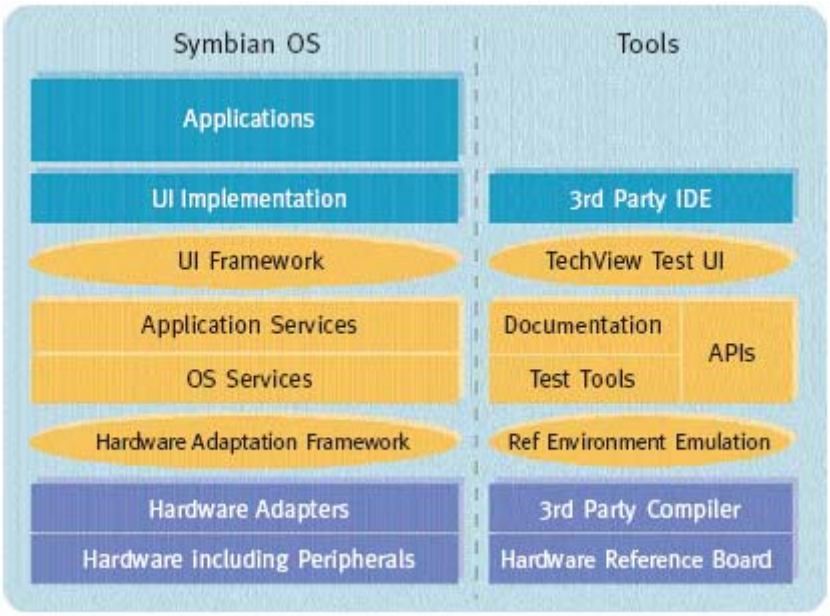
Tavoitteenani oli löytää jokaisen kehitysympäristön vahvat ja heikot puolet, sekä verrata niillä tapahtuvaa sovelluskehitystä toisiinsa. Henkilökohtaisena tavoitteenani on ollut uusien tekniikoiden ja mobiiliohjelmistokehityksen monimuotoisuuden oppiminen.

2 SYMBIAN-KÄYTTÖJÄRJESTELMÄ

Symbian OS on Symbian Ltd:n kehittämä käyttöjärjestelmä pienitehoisille, akkukäyttöisille ja vähäresurssisille laitteille. Symbian Ltd:n perustivat Ericsson, Matsushita, Motorola, Nokia ja Psion vuonna 1998. Käyttöjärjestelmä pohjautuu Psionin PDA-laitteitaan varten vuodesta 1989 alkaen kehittämään EPOC-käyttöjärjestelmään. Symbian OS on älypuhelimissa käytettävistä käyttöjärjestelmistä yleisin. Sitä lisensoivat maailman johtavat matkapuhelinvalmistajat, joilla on yli 85 prosentin osuus vuotuisesta mobiilipuhelimien myynnistä. /1/, /6/, /7/

Symbian OS on olioperustainen ja avoin käyttöjärjestelmä, joka tekee matkapuhelimista alustan sovelluskehitykselle ja palveluille, joita periaatteessa kuka tahansa voi kehittää. Laitevalmistajista riippumattomat ns. kolmannet osapuolet voivat luoda myös omia sovelluksiaan Symbian-laitteisiin. Symbian OS on komponenttipohjainen, pieni ja monipuolinen käyttöjärjestelmä, joka tukee mm. moniajoa. /1/, /7/

Symbian-käyttöjärjestelmään voidaan kehittää natiivisovelluksia C++-kielellä, sillä käyttöjärjestelmä on toteutettu Assembler-kielistä ydintä lukuun ottamatta täysin C++-kielellä. Symbian OS ei käytä kuitenkaan C++-kielen standardikirjastoja, vaan toteuttaa omat kirjastonsa näiden tilalle. Kirjastot on suunniteltu alusta lähtien kuluttamaan mahdollisimman vähän muistia ja ne ovat myös jossain määrin matalamman tason kirjastoja kuin standardit C++-kirjastot, mikä saattaa tehdä niiden käyttämisestä haastavaa. Java on ollut mukana Symbianin versiosta 5 saakka, ja tällä hetkellä Symbian-laitteissa on tuki J2ME:n MIDP-profiilin tai Personal Profilen mukaisille määrityksille, tai molemmille. C++- ja Java-kielten lisäksi Symbian-ohjelmointivälineiden joukko on laajentunut kattamaan mm. skriptikieliä, joista vahvimmin edustettuna on Python. Natiivisovellusten lisäksi Symbian-laitteisiin voidaan kehittää selainkäyttöliittymän kautta käytettäviä palveluja. /1/, /7/



Kuva 1 Symbian OS -käyttöjärjestelmän (versio 9.3) arkkitehtuurikuva /8/

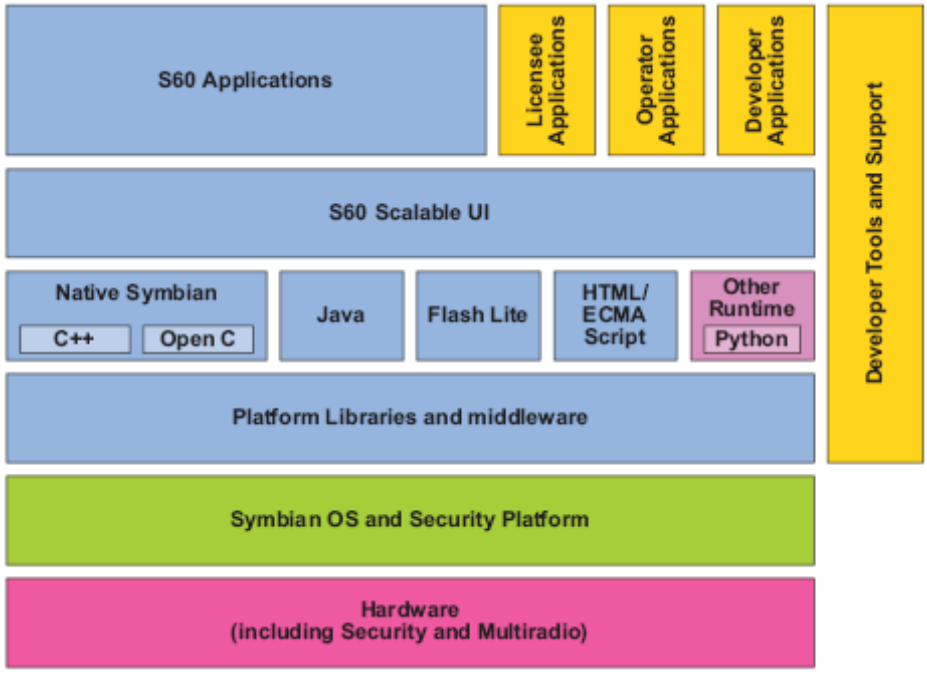
Symbian OS -käyttöjärjestelmässä Application Framework on sovelluskehys eli polymorfinen DLL, joka sisältää palvelut sovelluksille. Tämän pohjan päälle lisensoijat rakentavat oman sovitetun käyttöliittymänsä, kuten Ericssonin UIQ tai Nokian S60. Näillä Symbianin päälle rakennetuilla ympäristöillä on kaikilla oma SDK-pakettinsa, jonka avulla sovelluskehitys tehdään. Eräiden arvioiden mukaan mobiililaitteen Symbian-järjestelmässä 80 % lopputuloksesta tulee alkuperäisestä Symbian-järjestelmästä ja 20 % on räätälöinyt lisensoija. Nokian kehittämä S60 on Symbian-ympäristöistä selkeästi yleisin. /1/, /9/

3 S60-SOVELLUSALUSTA

S60 tunnettiin ennen nimellä Series 60, mutta vuonna 2005 Nokia muutti sen virallisen nimen lyhennettyyn muotoon S60. Nokian S60 tarjoaa monipuolisen ohjelmistoalustan älypuhelimille. Se tukee suuria värinäyttöjä ja sen suunnittelussa on yritetty huomioida käyttöliittymän helppous. S60-sovellusalusta hallitsee normaalit puhelintoiminnot ja myös vaativat kehittyneet sovellukset. S60:n avoimuuden ansiosta S60-puhelimille on tarjolla suuri määrä eri käyttötarkoituksiin soveltuvia yritysten ja yksityisten henkilöiden tekemiä sovelluksia. /6/, /10/

3.1 S60:n arkkitehtuuri /10/

S60-pohjaiset laitteet koostuvat eri tasoista (kuva 2). Toiminnallisuuden kautta S60 voidaan jakaa kolmeen päätasoon: sovellukset, käyttöliittymä ja välikerros (enablers). Nämä eri tasot ovat vastuussa S60-laitteiden eri toiminnoista.



Kuva 2 S60-sovellusalustan arkkitehtuurikuva /10/

S60 sisältää monia sovelluksia, jotka vaihtelevat monikäyttöisistä sovelluksista, kuten asetusten säätämisestä, yksikäyttöisempiin sovelluksiin, kuten kameraan. Sovelluksia voi tehdä useilla eri ohjelmointikielillä, kuten C++:lla, OpenC:llä, Pythonilla ja Javalla.

S60:n graafinen käyttöliittymä on maailmalla arvostettu ja hyväksi todettu. Se sijoittui toiseksi vuonna 2003 järjestetyssä iF Design Award Galassa, Communication / Digital Media -kategoriassa /10/. S60:n käyttöliittymä on helposti muunnettavissa erilaisten käyttäjien tarpeisiin muokattavuutensa ja skaalautuvuutensa ansiosta. S60:n käyttöliittymän tehtävä on antaa loppukäyttäjän navigoida ja käynnistää sovelluksia laitteessa.

S60 tukee monia eri teknologioita, joiden avulla on mahdollista toteuttaa tehokasta laitteen hallintaa, hallita tietokoneverkkoja tai paikallista liitettävyyttä sekä toteuttaa sovellusten ajamista ja hallintaa.

3.2 S60:n eri versiot /1/ /11/

S60 on kehittynyt rinnan Symbian-käyttöjärjestelmän kehityksen kanssa. Ensimmäinen S60-versio, S60 1st Edition, perustuu Symbian OS -versioon 6.1. Uusimmat puhelimet käyttävät S60 3rd Editionia, joka perustuu Symbian OS -versioon 9.1. S60 3rd Editionille on tullut myös uudempi versio Feature Pack 1:en myötä, jolle on juuri julkaistu älypuhelin, N95.

3.2.1 S60 1st Edition

Nokian julkaisema ensimmäisen versio S60:stä oli versio 0.9, joka perustuu Symbian OS 6.1:een. Tälle versiolle Nokia julkaisi vain yhden puhelimen, Nokia

7650:n (kuva 3). Tämä puhelin oli ensimmäinen matkapuhelin, jossa oli integroitu digitaalikamera ja Symbian OS -käyttöjärjestelmä. Se tuli myyntiin vuonna 2002. Pienten parannusten jälkeen Nokia julkaisi S60 1st Editionin, joka perustuu myös Symbian OS -versioon 6.1. S60 1st Editionille julkaistiin jo enemmän puhelimia, myös muilta valmistajilta.

Kuva 3 Ensimmäinen S60-puhelin, Nokia 7650 /12/

3.2.2 S60 2nd Edition

Nokia julkisti S60 2nd Editionin vuonna 2003. Tämä S60-versio pohjautuu Symbian OS -versioon 7.0s. Myöhemmin 2nd Editionin ominaisuuksia lisättiin kolmella Feature Packillä. Feature Pack 1 toi entistä paremman internetselaimen ja tuen EDGE-verkoille, Feature Pack 2 tuen WCDMA-verkoille ja Feature Pack 3 tuen entistä suuremmille näytön resoluutioille. Yhä useammat Nokian puhelimet rupesivat käyttämään S60-ympäristöä 2nd Editionin myötä.

3.2.3 S60 3rd Edition

Symbian OS:n versioon 9.1 perustuva S60 3rd Edition muutti olemassa olevaa Symbian- ja S60-sovelluskehitystä joiltakin osin merkittävästi. Uuden version myötä käyttöön tuli uusi kääntäjä, uusi kernel (EKA2) ja alustan turvallisuuden

uudellen toteutus. Uuden ympäristön Windows-emulaattorit käyttävät samaa ytimen koodia, joten myös emulointi saadaan aiempia versioita luotettavammaksi. 3rd Editionin myötä rikottiin binääriyhteensopivuus vanhoihin versioihin. Binääriyhteensopivuus rikottiin uuden turvallisuusjärjestelmän takia, joka vaatii kaikkien sovellusten olevan allekirjoitettuja. Allekirjoituksen tason mukaisesti sovellus saa erilaisia kykyjä, joilla se pääsee käsiksi käyttöjärjestelmän eri toimintoihin.

S60 3rd Edition puhelimia on julkaistu jo lukuisia. Nokialta on juuri tullut myyntiin N-sarjan lippulaiva N95 (kuva 4), joka on rakennettu S60, 3rd Edition Feature Pack 1:n päälle. Tämä puhelin edustaa tulevaisuuden matkapuhelinta, jossa on muun muassa integroitu GPS-toiminnallisuus.

Kuva 4 Nokian uusin lippulaiva, N95 /13/

4 J2ME

J2ME eli Java 2 Platform, Micro Edition, on Sun Microsystemsin kehittämän Java-teknologian kevyehkö sovellusympäristö, joka on tarkoitettu sulautettujen ja ominaisuuksiltaan rajoitettujen laitteiden, kuten puhelinten, ohjelmointiin. J2SE-alustaan verrattuna luokkakirjastot ovat rajoittuneemmat. J2ME:ssä on otettu huomioon esimerkiksi matkapuhelinten pelien näytön ja näppäimistön rajoitukset. /2/

Java-ympäristön ydin on tavukoodina annettua ohjelmaa suorittava virtuaalikone. Pohjimmiltaan virtuaalikone on vain käskyjä suorittava ohjelmistokomponentti. Sen ominaisuuksien käytön helpottamiseksi Java-ympäristöissä on määritelty konfiguraatioita ja profiileja. Profiili voidaan ajatella sovellusohjelmoijan ja Java-ympäristön toteuttajan väliseksi sopimukseksi, kun taas konfiguraatio voidaan ymmärtää laitevalmistajan ja Java-ympäristön toteuttajan väliseksi sopimukseksi, joka määrittelee tietyn laiteryhmän vähimmäisvaatimukset. Mobiiliympäristöön on suunniteltu CLDC-konfiguraatio ja MIDP-profiili. MIDP-profiililla mahdollistetaan ohjelmointi niin, että kaikki kyseistä profiilia käyttävät laitteet toimivat samoin samalla ohjelmointikoodilla. /2/

4.1 CLDC-konfiguraatio

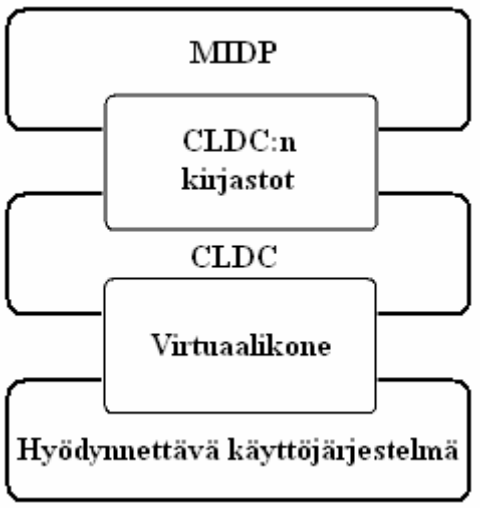
Connected limited device configuration (CLDC), joka tarkoittaa suomeksi yhteydellisen vaatimattoman laitteen konfiguraatiota, on Java-konfiguraatio, jossa mobiililaitteen erikoispiirteet on otettu huomioon. Konfiguraatio määrittelee minimitoiminnallisuuden, joka vaaditaan samaan kategoriaan kuuluvilta laitteilta. Konfiguraatiota voitaneen siis pitää sopimuksena J2ME-suoritusympäristön tekijän ja laitevalmistajan välillä. Sovelluskehittäjä ei yleensä näe CLDC:tä suoraan vaan CLDC:tä hyödyntävien profiilien kautta. /2/

CLDC sisältää tärkeimmät JLS:n (Java Language Standard) ja JVMS:n (Java Virtual Machine Standard) ominaisuudet, joukon olennaisimpia kirjastoja, internationalisointituen, joitakin turvallisuusominaisuuksia sekä I/O- ja verkkokirjastoja. CLDC sisältää Javan luokkien pienimmän mahdollisen osajoukon, jolla Javan virtuaalikone voi toimia. CLDC määrittelee myös muistinkulutuksen ja prosessoritehon maksimit ja minimit. /2/, /14/

Sovelluskehittäjälle profiilien hyödyntämä CLDC näkyy J2SE-ympäristön luokkakirjastojen typistettynä osajoukkona ja konfiguraation päälle rakennettujen profiilien kautta. /2/

4.2 MIDP

Mobile information device profile (MIDP) yhdistettynä CLDC:n kanssa on nykyisin yleisimmin käytetty Javan ajonaikainen ympäristö (runtime environment) matkapuhelimissa. Nokian S60-alustan puhelimet tukevat yleisesti juuri MIDP:tä. MIDP määrittelee peruskännyköille CLDC-konfiguraation päälle suunnitellun ohjelmointirajapinnan. Kuva (kuva 5) havainnollistaa konfiguraation ja profiilin välistä suhdetta. /2/



Kuva 5 Virtuaalikone, CLDC ja MIDP /2, s.91./

MIDP määrittelee ohjelmistosuunnittelijan käyttöön sovellusmallin, joka on nimeltään midlet. Lisäksi profiili määrittelee käyttöliittymän ohjelmoinnissa tarvittavan rajapinnan, jota voi hyödyntää MIDP-profiilin mukaisissa Java-sovelluksissa. /2/

5 Python

Python on monipuolinen, korkean tason tulkattava ohjelmointikieli, joka on alun perin kehitetty yhdistämään skriptikielten ja tavanomaisten ohjelmointikielten hyvät puolet. Pythonia on yleisesti pidetty helppona kielenä oppia yksinkertaisen syntaksinsa ja korkean tason tietorakenteidensa takia. Python-koodin koko on yleensä monta kertaa pienempi kuin vastaavan toteutuksen C++- tai Java-koodin koko. Python-tulkki ja -kirjastot on kehitetty avoimen lähdekoodin projektina, ja niitä levitetään Pythonin oman lisenssin alla, joka on yhteensopiva GPL-lisenssin kanssa. /4/, /15/

Python tukee monia lähestymistapoja ohjelmointiin. Pythonia voi käyttää mm. oliopohjaisena, rakenteellisena ja funktionaalisena ohjelmointikielenä. Python-koodin tulkitseminen ja ajoa edeltävä optimointi voivat joskus olla hitaita prosesseja. Sen vuoksi Pythonia ajetaan usein tavukoodina hieman Javan tapaan. Tulkkaamattoman Python-skriptin ajaminen saattaa olla monia kertoja C-kielisen ohjelman ajamista hitaampaa, vaikka suurimmat eroavuudet suorituskyvyssä johtuvat käytetyistä tietorakenteista ja algoritmeista. Lisänopeutta tarvittaessa Pythoniin voi lisätä C++-kielisiä laajennuksia. /4/, /15/

5.1 Python S60:n päällä

Vuonna 2006 Nokia julkaisi ensimmäisen version portatusta Pythonista S60:n päällä. Python-kielen normaalien ominaisuuksien lisäksi PyS60 antaa mahdollisuuden käyttää älypuhelinten uniikkeja ominaisuuksia, kuten kameraa, bluetoothia ja kalenteria. PyS60:n työkalut tarjoavat nopeaa sovelluskehitystä ja prototyyppien tekoa ja mahdollisuuden tehdä stand-alone S60-sovelluksia Python-kielellä. PyS60:n uusin versio on tällä hetkellä 1.3.20, joka on tarkoitettu käytettäväksi S60:n 2nd ja 3rd Editionien kanssa. PyS60 1.3.20 perustuu Python 2.2.2:een. /16/

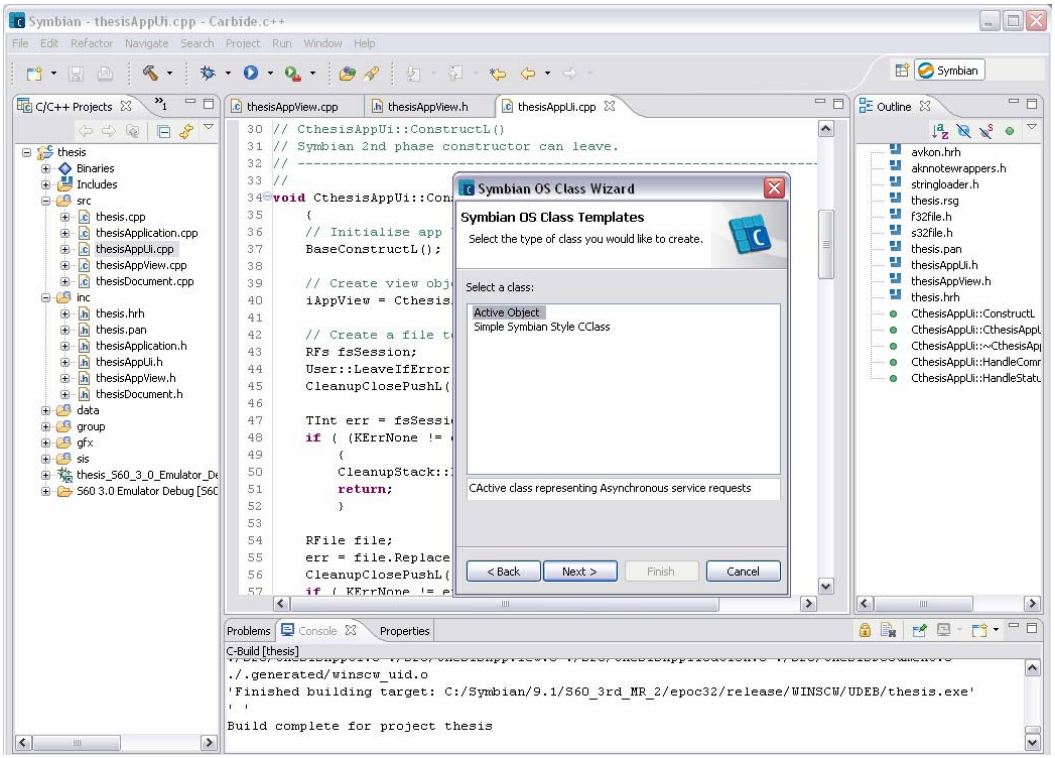
6 Kehitystyökalut

Symbian-ympäristössä on monia kehitystyökaluja erilaisiin käyttötarkoituksiin. Tehtäessä sovelluksia S60-laitteille tarvitaan Nokian toimittama SDK C++, Java tai Python kehitykselle. Useissa yrityksissä Symbian C++ kehitystä tehdään vielä Nokian CodeWarrior-työkaluilla, mutta Carbide-työkalut ovat varsinkin yksityiskäytössä valta-asemassa. Carbide.c++ tarjoaa kehitysympäristön C++-kehitykselle, ja Carbide.j taas kehitysympäristön J2ME-kehitykseen. Carbide pohjautuu avoimen koodin Eclipse-kehitystyökaluun. Nokian toimittamat SDK:t integroidaan toimimaan Carbide-kehitystyökalujen kanssa.

6.1 S60:n C++-kehitystyökalut

C++-kehitykseen S60-alustalla Nokia tarjoaa siis Carbide.c++ -työkalut. Niistä on tällä hetkellä julkaistu versio 1.1, joka on ilmainen kotikäytössä. S60-kehitykseen tarvitsee asentaa myös Nokian tarjoama SDK. SDK pitää valita aina sen matkapuhelimen S60-version mukaan, jolle haluaa kehittää sovelluksia. SDK:ta asennettaessa se integroituu automaattisesti Carbide.c++:saan, joten käyttäjän ei tarvitse huolehtia tästä. Seuraavan kerran, kun Carbide.c++ käynnistetään, ovat SDK ja työkalut valmiita käytettäväksi.

Carbide-työkalujen avulla voidaan mm. editoida koodia, luoda uusia luokkia, käynnistää emulaattori ja debugata koodia. Carbide osaa myös luoda uuden sovellusrungon ja pohjan uusille Symbian-luokille (kuva 6). Carbiden tarjoamat sovellusrungot ja uusien luokkien pohjat nopeuttavat uusien sovellusten luontia, kun ohjelmistosuunnittelija saa heti käyttöönsä valmiin ja kääntyvän sovellusrungon.



Kuva 6 Carbide.c++ Symbian C++-kehityksessä

Carbide.c++ -työkaluista on kehitteillä versio 1.2, joka tuo mm. UI Designerin C++ -kehittäjien avuksi. Tämä uusi versio on vasta beta-asteella, mutta UI Designer toimii jo hyvin muutamien testien perusteella. UI Designer nopeuttaa huomattavasti UI-komponenttien luontia, koska ohjelmistosuunnittelijan ei tarvitse kirjoittaa ohjelmakoodia, vaan UI Designer generoi sen muutamalla napin painalluksella. Symbian C++-kehitykseen on tarjolla myös monia muita työkaluja virheiden tarkasteluun.

6.2 PyS60-kehitystyökalut

Python-ohjelmien emulaattorilla ajamista varten tarvitsee S60 SDK:n päälle asentaa PyS60:n omat SDK-tiedostot. PyS60-paketin mukana tulee tarvittavat tiedostot, jotka lisätään S60 SDK:n epoc32-kansioon. Tiedostojen lisäyksen jälkeen Python-tulkin voi käynnistää ohjelmana emulaattorin sisällä. Python-tulkin (kuva 7) avulla voidaan ajaa Python-skriptejä ja käyttää konsolia. Python-skriptien

ajamiseen loppulaitteessa tarvitsee asentaa Pythonin ajonaikainen ympäristö ja PyS60 Script Shell.

Kuva 7 Python-tulkkia voi käyttää emulaattorissa

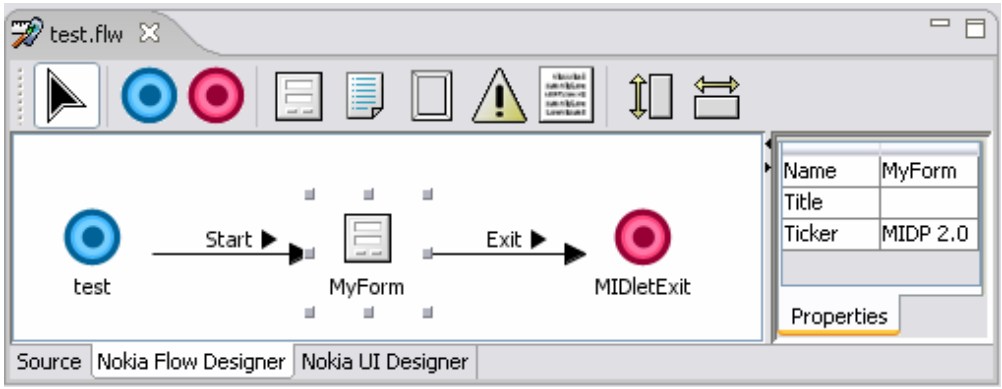
PyS60 tarjoaa sovelluskehittäjälle mahdollisuuden käyttää Pythonia puhelimessa Bluetooth-yhteyden avulla tietokoneen kautta. Käyttäjän tarvitsee vain käynnistää Bluetooth console -sovellus puhelimesta, jolloin se alkaa etsiä lähettyvillä olevia Bluetooth-laitteita. Konsolin käyttö on myös mahdollista TCP/IP-yhteyden ylitse. Yhteyden muodostamisen jälkeen käyttäjän on mahdollista käyttää konsolia suoraan tietokoneeltaan. Kaikki, mitä käyttäjä kirjoittaa tietokoneensa konsoliin, lähetetään puhelimelle ja tulkataan. Kaikki tulosteet lähetetään myös takaisinpäin ja kaikki poikkeukset (exception) tulostuvat myös tietokoneen konsoliin. Tämä helpottaa virheiden tarkastelua mm. GUI-sovelluksissa, koska ne peittävät Pythonin komentotulkin matkapuhelimessa. Tietokoneelle asennetun konsolin kautta voi ajaa myös erittäin hyödyllistä Pythonin debuggeria (pdb). /18/

PyS60:n työkalut tarjoavat mahdollisuuden nopeaan skriptien kehittelyyn, ajamiseen ja testaukseen muun muassa suoraan loppulaitteessa. Python-skriptit ovat usein hyvin pieniä ja niiden kirjoitus ja testaus on nopeaa, joten ohjelmistosuunnittelijan on helppo saada nopeasti loppulaitteessa toimivia sovelluksia.

6.3 MIDP Java -kehitystyökalut

Midlettien kehitykseen S60-alustalla käytetään yleisesti Eclipseä sekä sen päälle asennettavia Carbide.j-työkaluja ja S60 Symbian Java SDK:ta. Emulaattorin voi käynnistää Carbide.j:n kautta ja debuggaus onnistuu myös Carbide.c++:ssa opittuun tyyliin. Carbide.j tarjoaa ohjelmistosuunnittelijoille mahdollisuuden tehdä sovelluksia Flow Designerin ja Game Designerin avulla. Näitä designer-työkaluja käyttämällä ohjelmistosuunnittelijan ei tarvitse välttämättä aina edes koskea varsinaiseen ohjelmakoodiin.

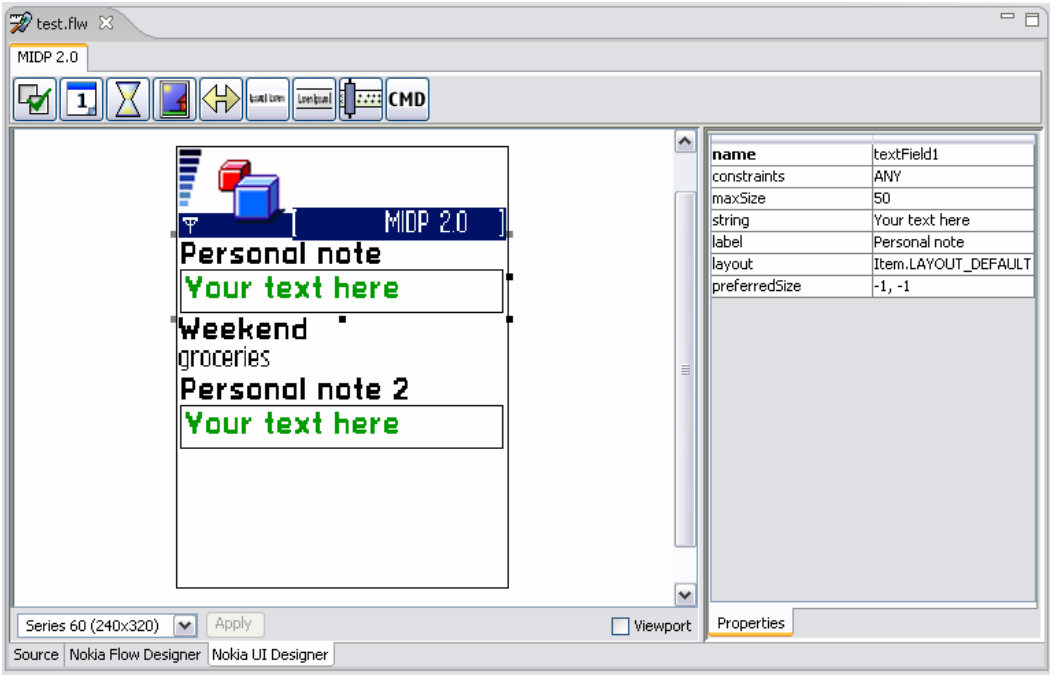
Flow Designerin avulla tehdään sovelluksia virtapainotteisesti (flow) (kuva 8). Ensiksi Flow Designerilla täytyy määrittää ns. alku- ja loppupisteet midletille, minkä jälkeen on mahdollista lisätä formeja, listoja ja tekstikenttiä midletiin. Flow Designerin avulla lisätyillä komponenteilla suunnitellaan midletille niin sanottu pohja.



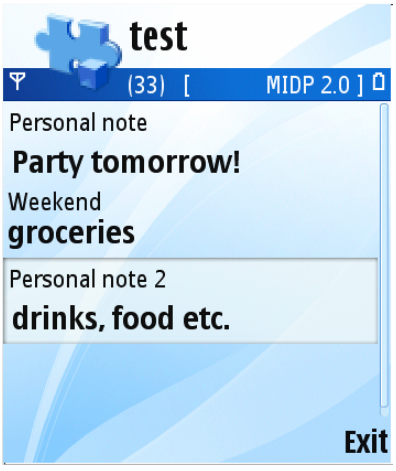
Kuva 8 Flow Designerilla on helppo tehdä pohja sovellukselle

Flow Designeriin liittyy myös UI Designer, jonka avulla on helppo lisätä komponentteja näkymiin (kuva 9). Komponenttien ja niiden sisältämien tietojen muokkaaminen onnistuu helposti komponentin omien asetusten avulla. UI Designerin toinen etu on, että se osaa näyttää välittömästi komponenttien lisäyksen

jälkeen lopputuloksen. Suunnittelijan ei siis tarvitse edes avata emulaattoria tarkistaakseen työnsä jäljen. Sovelluksen testaus voidaan suorittaa loppulaitteella tai emulaattorilla (kuva 10).



Kuva 9 UI Designerilla voi lisätä komponentteja näkymiin



Kuva 10 Testaus voidaan suorittaa emulaattorilla

Carbide.j:n tarjoamien tarjoamien designerien avulla on helppo luoda midlettejä ilman, että tarvitsee edes koskea koodiin. Nokian tarjoamien midlet-työkalujen avulla on helppo tehdä nopeasti toimivia UI-sovelluksia ja ne toimivat kaikissa puhelimissa, jotka tukevat kyseistä versiota MIDP:stä. Ohjelmistosuunnittelijat saavat näin helposti sovelluksensa monien käyttäjien ulottuville.

7 Virhetilanteiden ja poikkeusten käsittely eri alustoilla

Matkapuhelimet ovat lähentyneet merkittävästi kotitietokoneita viime vuosina, mutta niiden käytön erilaisuuden vuoksi mahdollisia ohjelmavirheitä täytyy ottaa eri tavoin huomioon. Kotitietokoneita käynnistetään yleensä päivittäin. Suljettaessa tietokone vapauttaa kaikki vanhat resurssinsa ja näin saadaan mahdolliset muistivuodot resurssoitua uudelleen käyttöön. Matkapuhelinten sovellusten, ja koko käyttöjärjestelmän, on taas tarkoitus olla suorituksessa pitkiä aikoja ilman keskeytystä ja uudelleen käynnistämistä. Niiden sovellusten pitäisi kyetä välttämään virhetilanteita ja tarvittaessa käsitellä virheet. Näiden edellämainittujen syiden ja matkapuhelinten pienemmän muistin takia matkapuhelinsovelluskehityksessä tulee ottaa erityisesti huomioon muistin käyttö ja sen vapauttaminen. Sovelluksen sulkeutuessa sen täytyy myös vapauttaa kaikki varaamansa muisti. /1/

7.1 Virhetilanteiden käsittely Symbian C++:ssa

Perinteisessä C++-ohjelmassa muistivirheet käsitellään yleisesti tarkastamalla NULL-osoittimet ehtolauseissa. Näiden tarkastuksen seurauksena tulee yleensä muutama lisäkoodirivi. ANSI C++:ssa poikkeusten käsittely tehdään try-catch-mekanismilla, joka taas kasvattaa koodin kokoa merkittävästi. Koodin koon kasvamisen seurauksena try-catch-mekanismi ei ole käyttökelpoinen pieniresurssisessa Symbian OS -järjestelmässä. Symbianin omat mekanismit soveltuvat paremmin matkapuhelinten muistinhallintaan niiden keveyden takia. Symbianissa poikkeustilanteet hoidetaan ns. Leave-mekanismilla. /1/

7.1.1 Leave-mekanismi

Poikkeus määritellään yleensä sovelluksen suorituksen aikaiseksi virheeksi, joka ei aiheudu suoraan ohjelmakoodin virheestä. Useita järjestelmässä mahdollisesti

tapahtuvia virheitä, kuten järjestelmän muistitilan puutetta, soketin avauksen epäonnistumista tai virhettä tiedostoa avatessa, ei pystytä ennakoimaan, eikä siten myöskään estämään ohjelmakoodissa. /1/

Symbian-järjestelmän Leave-mekanismin idea on yksinkertainen: jos muistia tai resurssia ei voida varata, koodissa syntyy Leave-tapahtuma. Tämä vastaa muiden ohjelmointikielten ja ympäristöjen throw-mekanismia. Poikkeuksen tapahtuessa funktion suoritus keskeytetään ja hypätään takaisin poikkeuksen käsittelyyn. Symbian-käyttöjärjestelmään on rakennettu oletuskäsittelijä poikkeuksille. Sitä käytetään, jos sovellus ei itse käsittele poikkeusta. /1/

Symbian-ympäristössä new-operaattori on ylikuormitettu Eleave-parametrilla, jonka ansiosta aiheutuu Leave-tapahtuma, jos muistia ei voida varata. Ohjelmistosuunnittelijoiden täytyy kuitenkin vapauttaa muisti normaalisti käyttäen delete-operaattoria. Symbianissa on tapana merkitä kaikki funktiot L-kirjaimella, jos niissä voi syntyä Leave-tapahtuma. Tämä näkyy myös esimerkissä (esimerkki 1) rivillä yksi. /1/

Koodissa voidaan myös itse aiheuttaa Leave-tyyppinen poikkeus käyttämällä User-luokan staattista Leave()-funktiota. Funktiolle annetaan tällöin integer-tyyppinen virhekoodi, kuten esimerkin (esimerkki 1) rivillä 6 on tehty. Parametrien tarkastus on myös mahdollista tehdä User-luokan staattisella LeaveIfNull()-funktiolla, kuten esimerkin (esimerkki 1) rivillä 3 on tehty. /1/

Useat API-funktiot palauttavat virhekoodin, jolla voidaan tarkastaa, onnistuiko haluttu toimenpide. Yleisiä tämänkaltaiset funktiot ovat R-luokissa, joissa käsitellään järjestelmän resursseja asiakas-/palvelinarkkitehtuurin mukaisen Client API:n kautta. Näissä funktioissa käytetään yleensä User-luokan LeaveIfError-funktiota, kuten esimerkin (esimerkki 1) rivillä 10 on tehty. /1/

```
1  void CFoo::MyFunctionL( MParam* aParam )
2      {
3      User::LeaveIfNull( aParam );
4
5      TInt error1 = KErrNoMemory;
6      User::Leave( error1 );
7
8      RFs fileServer;
9      TInt error2 = fileServer.Connect();
10     User::LeaveIfError( error2 );
11     }
```

Esimerkki 1 Leave-mekanismin käyttö

7.1.2 Poikkeusten käsittely

Symbian-käyttöjärjestelmä voi käsitellä Leave-tapahtumat automaattisesti, mutta erityistilanteissa sovelluskehittäjä voi käyttää TRAP- ja TRAPD-makroja. Niiden käyttöä pitäisi kuitenkin rajoittaa, koska ne lisäävät koodin kokoa. /1/

TRAP- ja TRAPD-makrojen käyttö on hyvin samanlaista. TRAPD-makrolle välitetään ensimmäisenä parametrina kokonaislukuparametri, jota ei määritellä etukäteen. Seuraavina parametreina annetaan lauseet, jotka voivat aiheuttaa Leave-tapahtuman. Suoritettavia funktiokutsuja voi olla useitakin, mutta tällöin Leave-tapahtuman syntyessä ei makron jälkeen ole tiedossa, missä funktiossa virhe tapahtui. Niinpä makrolla otetaan usein kiinni vain yhdessä funktiossa tapahtuva mahdollinen virhe. Makron suorituksen jälkeen ensimmäistä parametriä verrataan KErrNone –arvoon. Jos arvo on sama kuin KErrNone, ei virhettä ole tapahtunut. TRAP-makroa käytetään muuten samalla tavalla, mutta integer-tyyppinen ensimmäinen parametri täytyy esitellä etukäteen. Esimerkissä (esimerkki 2) on esitelty TRAPD-makron käyttäminen. /1/

```
1  TRAPD( error, DoSomethingL() );
2  if( KErrNone != error )
3      {
4      // Try to fix the error
5      }
```

Esimerkki 2 TRAPD-makron käyttö

7.1.3 Puhdistuspino

Symbian-laitteissa on tyypillisesti vähän muistia, ja laite voi olla ilman uudelleenkäynnistystä monia kuukausia. Muistivuotojen estämiseksi Symbianissa on käytössä mekanismi, joka tunnetaan nimellä puhdistuspino (Cleanup Stack). Puhdistusmekanismi toimii pääpiirteissään siten, että paikallisesti määriteltyjä osoittimia varten luodaan varmuuskopio mahdollisesti syntyvien Leave-tapahtumien varalle. Leave-tapahtuman sattuessa kaikki osoittimet tuhotaan automaattisesti. /1/

Puhdistuspinoa käytetään CleanupStack-luokan staattisten funktioiden kautta. Osoittimet laitetaan puhdistuspinoon käyttäen PushL-funktiota. Kahvat taas laitetaan puhdistuspinoon CleanupClosePushL()-funktion parametrina. Puhdistuspinosta voidaan poistaa objekteja Pop()-funktiolla. Jos instanssi tahdotaan tuhota tai kahva sulkea samalla, kun objekti poistetaan puhdistuspinosta, voidaan käyttää PopAndDestroy()-funktiota. Molemmille funktioille voidaan välittää kokonaislukuparametrina poistettavien objektien lukumäärä. Esimerkissä (esimerkki 3) on kuvattu CleanupStackin käyttöä luokan rakentamisessa. /1/

```
1 COwnClass* COwnClass::NewL( const TRect& aRect )
2     {
3     COwnClass* self = new ( ELeave ) COwnClass;
4     CleanupStack::PushL( self );
5     self->ConstructL( aRect );
6     CleanupStack::Pop( self );
7     return self;
8     }
```

Esimerkki 3 Puhdistuspinon käyttö

7.2 Virhetilanteiden käsittely PyS60:ssä

Pythonissa poikkeukset (exception) laukaistaan automaattisesti virheille. Laukaisu on myös mahdollista suorittaa itse omassa ohjelmakoodissa, jossa voidaan suorittaa

lisäksi virheiden sieppaus. Poikkeukset käsitellään kolmella komennolla (statement), joista ensimmäisellä on kaksi variaatiota: try/except ja try/finally. Kaksi muuta komentoa ovat raise ja assert. Try/exceptin (esimerkki 4) avulla siepataan (catch) ja yritetään palautua (recover) virheistä, jotka on nostettu itse tai Pythonin toimesta. Try/finallyn avulla suoritetaan siistimistoiminnot (cleanup), tapahtuipa poikkeus tai ei. Raise-komennon avulla voidaan laukaista poikkeus manuaalisesti koodista ja Assert-komennon avulla laukaista ehdonalainen poikkeus. /4/

```
1 try:
2     phone = bt_obex_discover()
3 except Exception, error:
4     note(unicode(error), 'error')
```

Esimerkki 4 Poikkeusten käsittely Pythonissa

Pythonilla kirjoitetuissa ohjelmissa poikkeuksia käytetään usein moniin tarkoituksiin, kuten virheiden käsittelyyn, tapahtumista (event) ilmoittamiseen, erikoistilanteisiin, lopetustoimenpiteisiin (termination) sekä epätavalliseen koodin ohjaukseen (unusual control-flow). /4/

PyS60:n API:t voivat lähettää minkä tahansa standardin Python-virheen. Tilanteissa, joissa palautetaan Symbianin virhekoodi, annetaan sen symboolinen nimi SymbianError -poikkeuksen arvoparametrinä. Jos funktioilla ei ole mitään erikoista virhekoodia palautettavanaan, ne palauttavat virheenään arvon None. /17/

Pythonin käyttöä ja virheiden tarkastelua varten on järkevää käyttää konsolia tietokoneelta Bluetooth-yhteyden yli, kuten kappaleessa 6.2 kerrottiin. Tällä tavoin poikkeukset ja virheet voi nähdä selkeästi tietokoneen näytöltä. Pythonissa, etenkin GUI-sovelluksissa, on yleensä järkevää ohjata virheiden tulostus erilliseen tiedostoon. /18/

7.3 Virhetilanteiden käsittely MIDP Javassa

Javan roskien keruun (garbage collector) ansiosta Javassa ei ole mahdollisuutta C++-tyylisille muistivuodoille, joissa oliolle varattu muisti jää vapauttamatta. Mobiilisovelluksia kehittäessä tulisi kuitenkin ottaa huomioon laitteiden vähäiset muistiresurssit ja jättää turhat muistivaraukset tekemättä, sillä varattu muisti on aina pois toisilta sovelluksilta. Midletien kehityksessä täytyy muistaa käyttää Javan normaalia poikkeusten käsittelyä try/catch-komennolla (esimerkki 5). /3/, /5/

```
1  try
2  {
3      hc = (HttpConnection)Connector.open(url);
4  }
5  catch (Exception e)
6  {
7      System.out.println("Error");
8  }
9  finally
10 {
11     if (hc != NULL)
12     {
13         hc.close;
14     }
15 }
```

Esimerkki 5 Poikkeusten käsittely Javassa

Virheiden käsittely on rajoitettua J2ME:ssä verrattuna J2SE:hen. CLDC määrittelee monia poikkeusluokkia, joita CLDC:n API:n metodit voivat heittää (throw). CLCD määrittelee myös ajonaikaisten virheiden käsittelyyn kolme luokkaa: Error, OutOfMemory ja VirtualMachineError. Nämä CLDC:n määrittämät virheet ovat vakavia, ja sovellusten ei tarvitse toipua niistä. Javan virtuaalikone voi antaa näistä virheen tai sulkea sovelluksen. Muut ajonaikaiset virheet käsitellään eri tavalla laitteesta riippuen. /3/, /5/

8 Komponentin toteutus eri alustoilla

C++, Java ja Python ovat ohjelmointikielinä hyvin erilaisia. Symbian C++ -kielellä tehdyt sovellukset ovat natiiviutensa ansiosta nopeampia kuin Java- tai PyS60-sovellukset, mutta varsin usein niiden kompleksisuus ja koodin määrä saattaa heikentää nopeudesta saatua hyötyä. Java-kieli tarjoaa useimmiten käyttäjälle valmiit luokat tarpeellisten toimintojen tekemiseen, joten ohjelmistosuunnittelijan tarvitsee tietää vain tarvittavat rajapinnat ja luokat, joita käyttää. Python-koodi on yleensä 1/3 tai 1/5 kertaa C++ tai Java-koodin kokoista /5/. PyS60-ympäristössä, jossa useat asiat on valmiiksi toteutettu, koodi voi olla jopa kymmeniä kertoja lyhyempää.

Bluetooth on tekniikkana nykyisissä älypuhelimissa tärkeässä osassa. Sitä käytetään muun muassa tiedonsiirtoon sekä puhelinten että puhelimen ja PC:n välillä ja myös hands-free-kuulokkeiden äänensiirtotekniikkana. Seuraavaksi työssä vertaillaan tiedostonsiirtoa Bluetoothin avulla käyttäen OBEX-tekniikkaa kaikilla näillä ohjelmointikielillä. OBEX tarjoaa menetelmän, jonka avulla on helppo lähettää tiedostoja laitteesta toiseen.

8.1 Tiedostonsiirto Bluetooth-tekniikalla Symbian C++ -ohjelmointikielellä

Toteutettaessa Symbian C++:n avulla Bluetoothia käyttäviä sovelluksia, ohjelmistosuunnittelijan täytyy tutustua julkisiin Bluetooth-ohjelmointirajapintoihin (API). Sekä S60 että Symbian tarjoavat erilaisia ohjelmointirajapintoja ohjelmistosuunnittelijan käyttöön. Tässä esimerkissä haetaan helpointa ja nopeiten ohjelmoitavaa tapaa siirtää tiedosto Bluetoothin avulla laitteesta toiseen laitteeseen. /21/

S60 tarjoaa kätevän välitysohjelmointirajapinnan, Send Ui:n, joka piiloittaa ohjelmakoodin yksityiskohdat ohjelmistosuunnittelijalta. Send Ui API:n avulla

ohjelmistosuunnittelija saa noin 30:llä itse kirjoitetulla rivillä toteutettua tarvittavan koodin Bluetooth-yhteyden yli tapahtuvalle tiedostonsiirrolle. Tässä täytyy kuitenkin ottaa huomioon, että Carbide.c++ on luonut tavallisen sovellusrungon S60 UI sovellukselle, johon oma ohjelmakoodi on lisätty. Liitteessä (liite 1) Send Ui:n käyttöön vaadittava koodi on toteutettu S60 UI -sovelluksen AppUi-luokkaan (BluetoothAppUi). Liitteessä on esitelty AppUi-luokkaan lisättävä metodi, jota kutsumalla voidaan tiedosto lähettää. Tarkemman lähdekoodin, jonka avulla voi SendUi:n käyttöä tutkia, voi ladata Forum Nokian sivuilta /22/. /21/

Bluetoothin yli tapahtuvaan tiedostonsiirtoon on tarjolla monia eri tapoja Symbian C++ ohjelmointikielellä S60-alustalla. Esimerkiksi Bluetooth-sockettien käyttö on suositeltavampaa, jos halutaan todellista kommunikaatiota laitteiden välille. Socketteja käyttämällä koodirivien määrä kasvaa kuitenkin valtavasti. Forum Nokia tarjoaa myös hyvän esimerkin jonka avulla voi paneutua tarkemmin OBEX-tekniikan käyttöön /22/. Send Ui ja sen käyttö on hyvä esimerkki siitä, kuinka helppoa hyödyllisten sovellusten tekeminen Symbian C++:lla voi parhaimmillaan olla.

8.2 Tiedostonsiirto Bluetooth-tekniikalla Python-skriptikielellä

Python-skriptien tekeminen on yleensä helppoa ja nopeaa. PyS60 tarjoaa monia valmiita funktioita, joiden avulla ohjelmistosuunnittelija voi ottaa käyttöön monia puhelimille ominaisia ominaisuuksia. PyS60:n avulla on helppo ottaa OBEX-tekniikka käyttöön ja siirtää tiedosto sitä käyttäen Bluetoothin avulla toiseen laitteeseen.

Kuten liitteestä (liite 2) näkyy, Python-kielen avulla tiedostonsiirto onnistuu muutamalla rivillä koodia. PyS60:n tarjoamien valmiiden OBEX-funktioiden avulla muodostetaan vain yhteys puhelinten välille ja siirretään tiedosto.

Tämän esimerkkikomponentin toteutus antaa osviittaa siitä, kuinka helppo Pythonin avulla on tehdä toimivia sovelluksia. Muutamalla rivillä koodia on mahdollista nopeasti tehdä toimivia sovelluksia, vaikka suoraan puhelimessa Bluetooth-konsolin avulla.

8.3 Tiedostonsiirto Bluetooth-tekniikalla Java-ohjelmointikielellä

Java tarjoaa monia eri Bluetooth-protokollia yhteyden muodostamiseen eri laitteiden välillä. J2ME:ssä on myös mahdollisuus käyttää OBEX-tekniikkaa tiedostonsiirtoon. /19/

Java-kielellä Bluetoothin yli tapahtuvaan tiedostonsiirtoon tarvitaan ohjelmointikoodia yllättävän paljon. OBEX-tekniikan käyttö ei suju niin vaivattomasti kuin Pythonilla tai Symbian C++:lla. Java-kielessä ohjelmistosuunnittelija ei pysty käyttämään OBEX-tekniikkaa helppojen funktioiden takaa kuten Symbian C++:lla ja Pythonilla.

OBEX-tekniikan käyttö vaatii Java-kielellä monia koodirivejä. Forum Nokia tarjoaa sivuillaan lähdekoodit esimerkkisovellukseen /20/, jossa siirretään tiedosto OBEX-tekniikan avulla. Tutkin OBEX-tekniikan käyttöä juuri tämän esimerkin avulla. Tämän esimerkin ja dokumentaation /19/ avulla OBEX-tekniikka on mahdollista liittää omiin sovelluksiinsa.

OBEX-tekniikan käyttöönotto oli huomattavasti vaikeinta Java-kielen avulla. OBEX-tekniikkaa käytetäänkin useimmin Java-kielen avulla PC-tietokoneiden ja puhelinten väliseen yhteydenpitoon, jossa tietokone toimii yleensä serverinä (J2SE-serveri).

9 Sovelluskehitys eri kehitysympäristöillä

S60-alustalle tehtävää sovelluskehitystä voidaan toteuttaa monella eri tavalla: ohjelmakoodin voi kirjoittaa kaikilla eri ohjelmointikielillä perinteistä tapaa käyttäen, mutta tarjolla on myös työtä helpottavia työkaluja. Varsinkin käyttöliittymäkehityksessä UI Designer -työkalu on erittäin hyvä apuväline. Symbian C++ -sovelluskehityksessä puolestaan apuna toimii Carbide.c++:n automaattisesti toteuttamat sovellusrungot ja luokkapohjat.

9.1 Sovelluskehitys Symbian C++ -ympäristössä

Symbian C++ on kielenä haastava myös ANSI C++:n osaajille, mutta useat ohjelmointirajapinnat, kuten Send Ui (kappale 8.1), nopeuttavat koodin kirjoittamista. C++-ohjelmistosuunnittelijoiden siirtyessä Symbian C++ -kehittäjiksi tulee heidän ottaa huomioon Symbianin omat mekanismit. Oppimiskynnystä lisää myös kielen erikoisuudet, kuten descriptorit ja aktiivioliot. Symbian C++ -kieltä opiskelevien kannattaisikin heti alusta pitäen pyrkiä käyttämään Symbian C++ -kielen ominaisuuksia ja pyrkiä välttämään ohjelmoimista perinteisen C++-kielen tapaan. Symbian C++ on kuitenkin näistä kolmesta kehitysympäristöstä ehdottomasti parhaiten dokumentoitu. Ennen varsinaisen koodin kirjoittamista on Symbian C++:lla tehtävässä ohjelmistokehityksessä tärkeää suunnitella tarkasti, miten komponentit kannattaa toteuttaa.

9.2 Sovelluskehitys MIDP Java -ympäristössä

Java tarjoaa käyttäjille yleensä valmiit luokat lähes kaikkeen toteutukseen. Tämä pätee myös MIDP Javassa, vaikka se onkin riisuttu versio J2SE:stä. Ohjelmistosuunnittelijoille onkin tärkeää tuntea tai ainakin osata etsiä Javan valmiit luokat, joiden avulla voidaan toteuttaa eri ohjelmistokomponentteja. Java-kieltä

käyttävien täytyy paneutua tarkasti J2ME:n rajoituksiin ja älypuhelimelle suunnattuihin luokkakirjastoihin. Nokian tarjoamat midlet-työkalut hoitavat lähes kaiken muun tarvittavan muutamalla napin painalluksella. Käyttöliittymäkehityksessä Carbide.j:n tarjoaman UI Designerin käyttö vähentää huomattavasti sovelluskehityksessä tarvittavaa aikaa.

9.3 Sovelluskehitys PyS60-ympäristössä

Python-skriptien tekeminen on nopeaa ja usein säästytäänkin monien koodirivien kirjoitukselta verrattuna C++- tai Java-kieleen. Pythonin ja PyS60:n tarjoamien valmisfunktioden avulla monien komponenttien, kuten Bluetooth-tekniikalla toteutettavan tiedostonsiirron (kappale 8.3), ohjelmoiminen hoituu jopa muutamissa minuuteissa, etenkin jos ohjelmistosuunnittelija tietää mitä valmisfunktioita käyttää. Toimivien Python-skriptien tekeminen on varsinkin kokeneille skriptien kirjoittajille nopeaa. Python-skriptejä tehtäessä varsinaisen suunnittelun voi joskus jättää esimerkiksi testisovelluksia tehtäessä kokonaan pois, koska niiden laatiminen on niin nopeaa. Python-sovellukset ovat huomattavasti hitaampia kuin Symbian C++-sovellukset. Tämä on varmasti yksi suurimmista syistä siihen, miksi laajempia sovelluksia ei Pythonilla yleensä tehdä.

10 Yhteenveto

Symbian C++, Python ja MIDP Java tarjoavat kaikki mahdollisuuden sovelluskehitykseen S60-alustalla. Ohjelmointikielenä Symbian C++ on haastavin, kun taas Python-skriptien tekeminen onnistuu ohjelmistosuunnittelijoilta muutamien tuntien opiskelun jälkeen. Pythonilla toteutetut sovellukset ovat kuitenkin huomattavasti hitaampia, kuin vastaavat C++-sovellukset. Tämä vaikuttaa siihen, että Python-sovellukset eivät aina tunnu varsinaisilta S60-sovelluksilta. Myös Java on kielenä suhteellisen helppo, mutta muista ohjelmointikielistä siihen siirtyminen vaatii valmiiden luokkien tutkimista ja hyväksikäyttämistä, sekä ohjelmistosuunnittelun muuttamista täysin oliopohjaiseksi.

Kehitystyökalut helpottavat suuresti sovelluskehitystä. Java-työkalut ovat tällä hetkellä hieman edellä Symbian C++ -kehitystyökaluja, lähinnä kehittyneen UI Designerin ansiosta. Pythonilla puolestaan tehdään kehitystä lähinnä tekstieditorilla sen skripti-luonteesta johtuen.

Mobiililaitteiden vähäisistä resursseista johtuen virheiden ja poikkeustilanteiden käsittely on tärkeää. Symbian C++ tarjoaa omat mekanisminsa ei-toivottujen tapahtumien, kuten muistivuotojen, varalle. Javalla tai Pythonilla ohjelmoitaessa ohjelmistosuunnittelijat eivät joudu käyttämään Symbian C++:n tyyppistä virheiden ja poikkeusten hallintaa, vaan ohjelmistosuunnittelijat käyttävät kielten omia käytäntöjä. Java- ja Python-kielen käyttäjät eivät siis joudu opettelemaan uusia mekanismeja.

Tutkimukseni tuloksena S60-alustalle tehtävä sovelluskehitys, joka toteutetaan ohjelmointikoodia kirjoittamalla, on helpointa Python-kielen avulla. Etenkin prototyyppien ja demojen tekeminen on suositeltavaa toteuttaa Pythonilla, jos asiakas esimerkiksi vaatii niitä nähtäväksi nopealla aikataululla. Käyttöliittymän

toteuttamisessa ja suunnittelussa puolestaan Carbide.j:n tarjoama UI Designer on tällä hetkellä lyömätön. Sen avulla on helppo tehdä S60-alustan tyylejä seuraavia sovelluksia varsin lyhyessä ajassa. Carbide.c++:n versio 1.2 tuo UI Designerin tarjolle myös Symbian C++ -kehitykseen, jolloin myös sillä tehtävä käyttöliittymäkehitys muuttuu nopeaksi. Symbian C++ -sovellusten etuna on niiden nopeus muihin kehitysympäristöihin verrattuna. Symbian C++ vaatii kuitenkin tällä hetkellä paljon ohjelmakoodia useimpien komponenttien laatimiseen, mikä saattaa vaikeuttaa kieleen tutustumattomien ohjelmistosuunnittelijoiden ohjelmoinnin aloittamista.

LÄHDELUETTELO

Painetut lähteet

1 Niskanen, Pekka, Symbian-ohjelmointi - tehokas hallinta. Readme.fi. Jyväskylä 2005.

2 Mikkonen, Tommi, Mobiiliohjelmointi. Talentum Media Oy. Jyväskylä 2004.

3 Allin, Jonathan, Wireless Java for Symbian Devices. John Wiley & Sons Ltd. Baffins Lane, Chichester, West Sussex, England 2001.

4 Lutz, Mark - Ascher, David, Learning Python, Second Edition. O'Reilly Media, Inc. United States of America 2004.

5 Digia, Programming for the Series 60 Platform and Symbian OS. Digia Incorporated. Helsinki 2003.

Sähköiset lähteet

6 Symbian OS. [www-sivu]. [viitattu 20.3.2007] Saatavissa: www.symbian.com

7 Wikipedia. [www-sivu]. [viitattu 19.4.2007] Saatavissa: http://fi.wikipedia.org/wiki/Symbian

8 Symbian. [pdf]. [viitattu 19.4.2007] Saatavissa: http://www.symbian.com/files/rx/file7999.pdf

9 S60 Home. [www-sivu]. [viitattu 19.4.2007]. Saatavissa: www.s60.com

10 S60. [pdf]. [viitattu 19.4.2007]. Saatavissa: http://www.s60.com/pics/pdf/S60_3rd_Ed_2007.pdf

11 Wikipedia. [www-sivu]. [viitattu 19.4.2007]. Saatavissa: http://fi.wikipedia.org/wiki/S60

12 Nokia Suomi. [www-sivu]. [viitattu 19.4.2007]. Saatavissa: http://www.nokia.fi/puhelimet/7650

13 Reg Hardware. [www-sivu]. [viitattu 19.4.2007]. Saatavissa: http://www.reghardware.com/2006/09/26/nokia_unveils_n95/

14 Wikipedia. [www-sivu]. [viitattu 19.4.2007]. Saatavissa: http://en.wikipedia.org/wiki/J2me

15 Wikipedia. [www-sivu]. [viitattu 19.4.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Python

16 Python for S60. [www-sivu]. [viitattu 19.4.2007]. Saatavissa: http://wiki.opensource.nokia.com/projects/Python_for_S60

17 PyS60. [pdf]. [viitattu 19.4.2007]. Saatavissa: http://www.mobilenin.com/pys60/resources/PythonForS60_doc_1_3_14.pdf

18 PyS60. [pdf]. [viitattu 19.4.2007]. Saatavissa: http://wiki.opensource.nokia.com/images/8/8a/ Programming_with_PyS60_1_2.pdf

19 Resource Information. [pdf]. [viitattu 20.4.2007]. Saatavissa: http://sw.nokia.com/id/947942ec-8c77-4f56-b4cf-b4dd6764fdfc/MIDP_OBEX_API_Developers_Guide_v1_0_en.pdf

20 Resource Information. [www-sivu]. [viitattu 20.4.2007]. Saatavissa: http://www.forum.nokia.com/info/sw.nokia.com/id/b37cb6fa-fa2b-429c-9d1e-3b9728a23ac8/MIDP_Bluetooth_OBEX_Example_Business_Card_Exchange_v1_0_en.zip.html

21 Resource Information. [www-sivu]. [viitattu 20.4.2007]. Saatavissa: http://www.forum.nokia.com/info/sw.nokia.com/id/125b7ff5-f2dd-4441-8cfe-59e23c006373/MIDP_Bluetooth_API_Developers_Guide_v2_0_en.pdf.html

22 Resource Information. [www-sivu]. [viitattu 20.4.2007]. Saatavissa: http://www.forum.nokia.com/info/sw.nokia.com/id/385f811a-2940-444f-b906-e5d3444e121c/S60_Platform_Bluetooth_OBEX_Example_v1_0_en.zip.html

```
void CBluetoothAppUi::SendFileViaSendUiL()
    {
    TFileName path;
    path.Append(KOwnFileName);

    TSendingCapabilities capabs( 0, 1024, TSendingCapabilities::ESupportsAttachments );

    RFs fs;
    CleanupClosePushL(fs);
    User::LeaveIfError( fs.Connect() );
    fs.ShareProtected();

    TInt error( KErrNone );

    RFile temp;
    User::LeaveIfError( temp.Open( fs, path, EFileShareReadersOnly | EFileRead ) );
    CleanupClosePushL(temp);

    CMessageData* messageData = CMessageData::NewL();
    CleanupStack::PushL(messageData);
    messageData->AppendAttachmentHandleL(temp);

    TRAPD(err, iSendUi->ShowQueryAndSendL(messageData, capabs) );

    if( err ) // error occured during sending
        {
        // Send some info to user..
        }

    CleanupStack::PopAndDestroy(messageData);
    CleanupStack::PopAndDestroy(&temp);
    CleanupStack::PopAndDestroy(&fs);
    }
```

```
# bluetooth.by

from appuifw import *
from e32socket import *

try:
    phone = bt_obex_discover()
    file = query(u'File Selection', 'text')
    bt_obex_send_file(phone[0], phone[1].values()[0], file)
    note(u'File Sent')
except Exception, error:
    note(unicode(error), 'error')
```